I use this mini-project to get a sense for the way you look at a problem and also how you code. While not all research projects in my group require lots of coding, many do. Do your best to complete the three subparts below. You may use any books or online resource, but please do this on your own. Submit your code (you can use any programming language) as well as a short PDF document summarizing your solution (including plots) to the questions below.

Gradient Descent Primer

Suppose that we would like to find a local minimum of a function $f(\boldsymbol{x}), \boldsymbol{x} \in \mathbb{R}^n$, and $f : \mathbb{R}^n \to \mathbb{R}$. An iterative algorithm starts at an initial point, say x_0, and generates a series of points (iterates) $x_1, x_2, \ldots, x_n$, where the last point x_n is either a (local) minimizer or very close to one. The points are usually generated as:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k, \qquad k = 0, 1, 2, \ldots \tag{1}$$

where $\alpha_k \in \mathbb{R}$ is the step length and $\boldsymbol{d}_k \in \mathbb{R}^n$ is the step direction in the k^{th} iteration. The step length, α_k, can be chosen as a fixed value or one that minimizes the function for that iteration. If the step directions are chosen as $\boldsymbol{d}_k = -\nabla f(\boldsymbol{x_k})$, then the algorithm is referred to as the *steepest descent method*.

1. Implementing the steepest descent algorithm to a differentiable function is straightforward and requires the following initial steps:
   (a) Determine the function to be minimized.
   (b) Determine the gradient of the function.
2. The steepest descent algorithm can then be implemented in the following way:
   (a) Choose an initial guess, x_0 and set your iteration counter $k = 0$.
   (b) Perform the following loop until a convergence criterion (stopping criterion) is met:
      i. Calculate the gradient at the current point.
      ii. Determine the step length in the direction of the gradient.
      iii. Find the next point $\boldsymbol{x}_{k+1}$ by using Equation (1).
      iv. Increment the iteration counter k by one.
3. When the convergence criterion has been satisfied, the final point $\boldsymbol{x}_n$ should be such that $f(\boldsymbol{x}_n)$ is a local minimum of your function or close to one.

The steepest descent algorithm is a useful tool, however, it may be very slow for some problems. Although more sophisticated versions of this algorithm are used in practice to overcome these issues, the major principles of the method remain similar.

Activity 1: Gradient Descent

Let's take a look at the problem of predicting your total course grade in a class based on how well you did on the midterm exam. We can start by looking at the data from 6 of last year's students:

Student	A	B	C	D	E	F
Midterm Grade	92	55	100	88	61	75
Course Grade	95	70	95	85	75	80

Assume that there is a linear relation between the dependent variable (the Course Grade) and the independent variable (the Midterm Grade). Write a gradient descent program to find the best fitting linear relation for the data. Similar to least squares, we would like to minimize the error between the actual course grade values and the ones predicted using the linear relation. Have your program display a plot of the best fit line along with the raw data. Put the equation of the best fit line and your plot into your PDF summary.

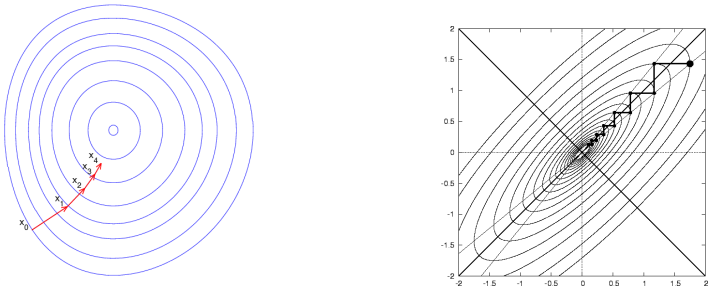

Figure 1: (Left) A possible path of a variable as it follows the gradient each iteration. (Right) Two parameters of significantly different magnitudes create a situation in which the gradient descent method uses significantly more iteration steps. The contours of the cost function are shown along with the steps of the gradient method.

Activity 2: Feature Scaling

Although the gradient method is a simple algorithm, you most likely already found a few aspects of the process that were a bit tricky. In particular, you probably noticed that finding a step size that worked well took some trial and error. Part of this is due to the fact that the step size is multiplied by the derivative in order to update the parameter value. Therefore, the value of the function itself (and the derivative) influences how big or small the step size should be. The gradient method performs poorly when one parameter is significantly larger than another, such as in the last activity where the y-intercept is much larger in value than the slope. Figure 1b shows how a function with differently scaled parameters exhibits stretched level curves. Imagine trying to use a gradient method with a relatively large step size - you can see it would diverge quite quickly.

One way to avoid this difficulty is to rescale the problem so all of the parameters are on roughly similar scale. This makes the contours in Figure 1b more circular than elliptical. We accomplish this by rescaling the data using $\tilde{x} = \frac{x-\bar{x}}{\sigma_x}$ and $\tilde{y} = \frac{y-\bar{y}}{\sigma_y}$, where $\bar{x}$ is the mean of the x values and σ_x is the standard deviation of the x values (note that this is not the only way to normalize the parameter values). Running the gradient method on a normalized problem is significantly easier and more robust to your choice in step size.

Update your code from Part 1, this time normalizing the midterm and course grades first. Keep in mind that the slope and y-intercept will change after you normalize your data, so you will have to use a reverse transformation to "unnormalize" these slope and y-intercept values. Again show the plot of the "unnormalized" best fit line and the raw date in your PDF summary, as well as the equation of the best fit line. For the same step size and convergence criteria does the normalized problem converge in fewer iterations?

Activity 3: Extended Gradient Descent

In many cases, the quantity we measure is dependent on more than a single independent variable. Let's look at another problem in which we analyze the effect of weight (kg), age (years), and stress (stress index, 0-100) on blood pressure (BP).

Person	1	2	3	4	5	6	7	8	9	10	11	12	13
Weight	69	83	77	75	71	73	67	71	77	69	74	86	84
Age	50	20	20	30	30	50	60	50	40	55	40	40	20
Stress	55	47	33	65	47	58	46	68	70	42	33	55	48
BP	120	141	124	126	117	129	123	125	132	123	132	155	147

Assume a linear model for predicting blood pressure by weight, age, and stress. Make a copy of your gradient descent program and update it for this larger model. Solving this problem, due to the extra dimensions (of different magnitues), is significantly easier if you first normalize the data!

Find the coefficients for the linear model and report the equation of the line in your summary. Create a vector with the predicted blood pressure values based on your model and plot the actual (y-axis) versus the predicted (x-axis) values. Also plot the line $y = x$. Put this plot in your summary as well.